Title
Methods of Simulating Networks from Exponential Random Graph Models

Author
Yang, Suoyi

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Methods of Simulating Networks from Exponential Random Graph Models

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Suoyi Yang

2020

ABSTRACT OF THE THESIS

Methods of Simulating Networks
from Exponential Random Graph Models

by

Suoyi Yang
Master of Science in Statistics
University of California, Los Angeles, 2020
Professor Mark Stephen Handcock, Chair

The exponential random graph models are a family of statistical models that are often used for analyzing networks. The most common method of simulating networks from these models is using the Metropolis-Hastings algorithm. However, the Metropolis-Hastings algorithm has certain disadvantages such as long mixing times and issues of degeneracy. As a result, the goal of this paper is to look for an alternative method of simulating networks from exponential random graph models that can improve upon the drawbacks of the Metropolis-Hastings algorithm. The three alternative methods explored in this paper are the Swendsen-Wang algorithm, the Gibbs sampler with Swendsen-Wang algorithm, and the iterative sampling with spectral clustering algorithm. Out of these three, the iterative sampling with spectral clustering ultimately proved to be the most viable algorithm. The rest of the paper focuses on analyzing the network simulations generated using this method.

The thesis of Suoyi Yang is approved.

Erin K. Hartman

Frederic R. Paik Schoenberg

Mark Stephen Handcock, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Exponential random graph models (ERGMs) are a family of statistical models that are commonly used to represent the stochastic processes that generate the ties between pairs of nodes in a network. In recent years, they have been used in a wide range of disciplines including sociology [7], biology [12], and political science [3] to name a few.

The most common method of simulating networks from ERGMs is to use the Metropolis-Hastings algorithm [9]. However, the Metropolis-Hastings algorithm can have certain disadvantages such as slower mixing times and issues with degeneracy [8]. Thus, this paper will look at three alternative methods of simulating from ERGMs. The first is the Swendsen-Wang algorithm, which is a well-established algorithm often used with Potts models [13]. An algorithm integrating the Gibbs sampler with Swendsen-Wang is developed in this paper as another potential method, and the iterative sampling with spectral clustering algorithm is created as the third alternative method.

Chapter 2 of this paper will focus discussing what an ERGM is and some key terms and concepts associated with it. From there, Chapter 3 will discuss how the Metropolis-Hastings algorithm is commonly used to simulate networks from ERGMs and also touch on some disadvantages of the method.

Chapter 4 will propose the Swendsen-Wang algorithm and a combination of Gibbs sampling and Swendsen-Wang as potential alternative methods of simulating from ERGMs. The chapter will also discuss the big issues that came up when attempting to implement these methods. To remedy some of the difficulties presented by the Swendsen-Wang algorithms,

Chapter 5 proposes an iterative sampling with spectral clustering method. As this algorithm proves to be the most viable of the three proposals, Chapter 6 will apply it to well known networks such as Sampson and Lazega to analyze its performance in simulating tie density and homophily. Chapter 7 will use the method look at inferences of the ERGM. Finally, Chapter 8 discusses the findings and thoughts on all three methods.

CHAPTER 2

Overview of Exponential Random Graph Model

Before attempting to simulate networks from the exponential random graph model, it is imperative to first have a good understanding of its notations and specifications.

Suppose a network with n nodes has an adjacency matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$ with entries y_{ij} for $i, j \in \{1, 2, ..., n\}$, where

$$y_{ij} = \begin{cases} 1 & \text{if there is an (un)directed tie from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}.$$

In addition, let the support of $\boldsymbol{Y}$ be denoted as $\mathcal{Y}$, which is the set of all possible and valid adjacency matrices. In this paper, $\mathcal{Y}$ is specified as the subset of all $n \times n$ matrices with elements of only ones and zeros and whose diagonal elements consists only of zeros. Since $y_{ij} = 0$ and $y_{ij} = 1$ indicate the lack and presence of a tie from i to j respectively, having diagonal entries of all zeros indicates that loops are not allowed in the network. More specifically, if only undirected networks are being considered, then $\mathcal{Y}$ is further limited to only contain symmetric matrices.

2.1 Model

The probability of a certain network occurring depends on $\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})$ where $\boldsymbol{\theta} \in \mathbb{R}^m$ is a vector of the model's coefficients and $\boldsymbol{g}(\boldsymbol{y})$ is an m-vector of statistics (number of ties, triangles, stars, etc. in the network) based on adjacency matrix $\boldsymbol{y} \in \mathcal{Y}$ [5]. In order to transform the

term into a probability, we can take the exponent of $\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})$ and divide it by a normalizing factor $\kappa(\boldsymbol{\theta}, \mathcal{Y})$. This ensures the result is positive and integrates to 1.

Thus, the distribution of $\boldsymbol{Y}$ can be parameterized as

$$\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(\boldsymbol{Y} = \boldsymbol{y}) = \frac{\exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})\right)}{\kappa(\boldsymbol{\theta}, \mathcal{Y})} = \frac{\exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})\right)}{\sum_{\boldsymbol{z}\in\mathcal{Y}} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{z})\right)}\,. \tag{2.1}$$

If covariates $\boldsymbol{X}$ are introduced into the network, then $\boldsymbol{g}(\boldsymbol{y})$ can be denoted as $\boldsymbol{g}(\boldsymbol{y}, \boldsymbol{X})$ instead.

2.2 Change Statistic

An important concept in ERGMs is the idea of *change statistics* [6], denoted $\boldsymbol{\delta}_{\boldsymbol{g}}(\boldsymbol{y})_{ij}$, or $\boldsymbol{\delta}_{\boldsymbol{g}}(\boldsymbol{y}, \boldsymbol{X})_{ij}$ if there are covariates $\boldsymbol{X}$ in the network.

First, to introduce some notations:

- $\boldsymbol{y}_{ij}^+$: network with $y_{ij} = 1$, keeping the rest of the network the same as before
- $\boldsymbol{y}_{ij}^-$: network with $y_{ij} = 0$, keeping the rest of the network the same as before
- $\boldsymbol{y}_{ij}^c$: rest of the network other besides y_{ij}

Then the vector of change statistics can be defined as

$$\boldsymbol{\delta}_{\boldsymbol{g}}(\boldsymbol{y})_{ij} = \boldsymbol{g}(\boldsymbol{y}_{ij}^+) - \boldsymbol{g}(\boldsymbol{y}_{ij}^-). \tag{2.2}$$

We can therefore see that the change statistic is essentially the change in the value of $\boldsymbol{g}(\boldsymbol{y})$ (defined previously in section 2.1) if the value of y_{ij} is changed from 0 to 1, but $\boldsymbol{y}_{ij}^c$ (rest of the network $\boldsymbol{y}$) is kept exactly the same.

Using change statistics, an alternative specification for the model in equation 2.1 can be obtained [6]:

$$\text{logit}[\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(Y_{ij}=1|\boldsymbol{Y}_{ij}^c=\boldsymbol{y}_{ij}^c)]=\log\frac{\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(Y_{ij}=1|\boldsymbol{Y}_{ij}^c=\boldsymbol{y}_{ij}^c)}{1-\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(Y_{ij}=1|\boldsymbol{Y}_{ij}^c=\boldsymbol{y}_{ij}^c)}=\boldsymbol{\theta}^\top\boldsymbol{\delta}_{\boldsymbol{g}}(\boldsymbol{y})_{ij} \tag{2.3}$$

From the equation above, the interpretation of vector $\boldsymbol{\theta}$ is clarified. Each component i of $\boldsymbol{\theta}$ denotes the increase in the conditional log-odds of the network if we were to increase the i-th component of $\boldsymbol{g}(\boldsymbol{y})$ by one unit. This increase in the components of $\boldsymbol{g}(\boldsymbol{y})$ results from changing the value of y_{ij} from 0 to 1, but keeping $\boldsymbol{y}_{ij}^c$ (rest of the network $\boldsymbol{y}$) exactly the same.

2.3 Important Terminologies

In order to have a full understanding of ERGMs and networks in general, there are several important terms that should be defined:

(1) **Exogenous** group membership is not affected by other variables in the system and is a measurable feature of the node (e.g., age, sex, height, etc.).

(2) **Endogenous** group membership is influenced by other factors in the system and is not a measurable feature of the node. In this case group membership is created by a "latent cluster" that is related through the relationships in the network.

(3) **Dyads** are random variables that indicate whether there is a tie between two nodes within a network. In an undirected network, the relationship between two nodes is always reciprocal and thus a dyad describing the relationship between nodes i and j can be denoted as a single random variable Y_{ij}. Conversely, the relationship between two nodes in a directed network is not always reciprocal. In this case, a dyad between two nodes can be denoted as a pair (Y_{ij}, Y_{ji}).

(4) **Dyadic independence term** is a term in an ERGM where we can find the value of $\boldsymbol{\delta}_{\boldsymbol{g}}(\boldsymbol{y})_{ij}$ without knowing anything about $\boldsymbol{y}$, with exception of perhaps the value of y_{ji}.

(5) **Dyad independent ERGM** is an ERGM consisting only of dyadic independence

terms. In this case

$$\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(Y_{ij} = 1 | \boldsymbol{Y}_{ij}^c = \boldsymbol{y}_{ij}^c) = \mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(Y_{ij} = 1)$$

(6) **Homophilous ties** are ties that occur between two nodes that share the same nodal attribute value (such as same age group, sex, or height group).

(7) A **connected component** of an undirected network is a subgraph of the network where any two nodes in the subgraph are connected together by at least one path. No nodes in a connected component should have any connection to nodes outside of it.

2.4 Simulations

Now that we have a better grasp of exponential random graph models, we can simulate networks from them. A big issue that presents itself when attempting to simulate from ERGMs is that the normalizing constant $\kappa(\boldsymbol{\theta}, \mathcal{Y})$ is extremely difficult to calculate. This is because even with just 30 nodes in the network, there are 2^{435} possible networks to sum over. The following chapters will present potential simulations methods that circumvent this issue.

CHAPTER 3

Metropolis-Hastings Simulation

Markov chain Monte Carlo (MCMC) methods can be used to simulate networks from ERGMs without having to calculate $\kappa(\boldsymbol{\theta}, \mathcal{Y})$ [9]. The main objective in using MCMC methods to simulate from ERGMs is to construct a Markov chain on $\mathcal{Y}$. To do this, take a network in $\mathcal{Y}$ and make Markov transitions from network to network until it converges to $\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y)$, the equilibrium distribution (or stationary distribution) in this scenario. After reaching this convergence, all the subsequent transitions can be taken as samples to form a sample of the desired network. The specific details of this method will be explored in the next section.

3.1 Method

A popular MCMC method used in the simulation of networks is the Metropolis-Hastings Algorithm [9].

Suppose the current network is denoted $\boldsymbol{y}_{\text{current}}$, and we propose a new network denoted as $\boldsymbol{y}_{\text{proposed}}$. Then according to the Metropolis-Hastings algorithm, a choice can be made on whether or not to transition from $\boldsymbol{y}_{\text{current}}$ to $\boldsymbol{y}_{\text{proposed}}$. The decision to transition to $\boldsymbol{y}_{\text{proposed}}$ occurs with probability

$$\min\left(1, \frac{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{proposed}})}{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{current}})}\right) \tag{3.1}$$

Note that the normalizing constant $\kappa(\boldsymbol{\theta}, \mathcal{Y})$ will cancel out in the fraction, which will greatly simplify the calculations. In a more general case, the $\boldsymbol{y}_{\text{proposed}}$ previously mentioned

can be generated by the Metropolis-Hastings algorithms by choosing it from a distribution dependent on $\boldsymbol{y}_{\text{current}}$.

Denote $q(\boldsymbol{y}_1, \boldsymbol{y}_2) = \mathbb{P}(\boldsymbol{y}_{\text{proposed}} = \boldsymbol{y}_2 | \boldsymbol{y}_{\text{current}} = \boldsymbol{y}_1)$. Then equation 3.1 can be rewritten as

$$\min\left(1, \frac{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = \boldsymbol{y}_{\text{proposed}})}{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = \boldsymbol{y}_{\text{current}})} \frac{q(\boldsymbol{y}_{\text{current}}, \boldsymbol{y}_{\text{proposed}})}{q(\boldsymbol{y}_{\text{proposed}}, \boldsymbol{y}_{\text{current}})}\right) \tag{3.2}$$

Note that the above ratio of ERGM probabilities can be simplified to

$$\frac{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{proposed}})}{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{current}})} = \frac{\exp(\boldsymbol{\theta}_0^\top \boldsymbol{g}(\boldsymbol{y}_{\text{proposed}}))}{\exp(\boldsymbol{\theta}_0^\top \boldsymbol{g}(\boldsymbol{y}_{\text{current}}))} = \exp\left(\boldsymbol{\theta}_0^\top [g(\boldsymbol{y}_{\text{current}}) - g(\boldsymbol{y}_{\text{current}})]\right) . \tag{3.3}$$

In addition, if the only difference between the proposed and current network is the change of a single tie, then [6]:

$$\frac{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{proposed}})}{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y} = y_{\text{current}})} = \exp\left(\boldsymbol{\theta}_0^\top \left[\pm \boldsymbol{\delta}_g(y)_{ij}\right]\right) \tag{3.4}$$

3.2 Disadvantages

While the Metropolis-Hastings algorithm does a good job of simulating from ERGMs, the method does have a few disadvantages. For example, if the initial starting network in $\mathcal{Y}$ is in a region of low density, then the initial samples may follow a very different distribution, and a large burn-in may be needed before samples can be drawn. In addition, the MCMC algorithms used to simulate networks from Markov graph models can have a tendency to converge inconsistently or to degenerate graphs (graphs that are either empty or complete) [8]. Issues can also arise if a step-size either too big or too small is used in the algorithm, as it could often lead to a slow-mixing Markov chain [9].

CHAPTER 4

Swendsen-Wang Simulation

Instead of the Metropolis-Hastings algorithm, we can consider using the Swendsen-Wang algorithm to simulate networks from ERGMs. This algorithm lends itself to a faster mixing time, which was one disadvantage of the Metropolis-Hastings algorithm mentioned in section 3.2. In addition, there are many similarities between Metropolis-Hastings and Swendsen-Wang (discussed further in the next section) which allow the latter to deal with the issue of the normalizing constant in a similar manner as the former.

4.1 Swendsen-Wang Method

Swendsen-Wang is typically applied to the Potts model, which is an $n \times n$ lattice with 4 nearest neighbor connections [2]. Each of the n^2 vertices in the Potts model has a state variable x_k for $k \in \{1, 2, ..., n^2\}$, which has a finite L numbers of labels. Letting $\boldsymbol{X}$ be a vector consisting of the label of each state variable, the Potts model is a Markov random field

$$\mathbb{P}(\boldsymbol{X}) = \frac{1}{Z} \exp\left(- \sum_{\langle k,\ell \rangle \in E} \beta_{k\ell} \mathbb{I}(x_k \neq x_\ell) \right) . \tag{4.1}$$

In the case of ERGMs, the Potts model is essentially our adjacency matrix $\boldsymbol{Y}$. More specifically, the $n \times n$ adjacency matrix $\boldsymbol{Y}$ will be our Ising model, which is a Potts model with only 2 labels $x_k \in \{0, 1\}$ [2]. The vertices of the Ising model represents the elements of

the adjacency matrix, while the state variables of the vertices are the *values* of the elements $y_{ij} \in \{0, 1\}$.

Now we introduce a new a set of variables for the edges in the Ising model. Let E be the set of all possible edges in the model. Then,

$$U = \Big\{ \mu_e : \mu_e \in \{0, 1\}, \forall e \in E \Big\}.$$

where edge e is "turned off" if and only if $\mu_e = 0$. μ_e is a Bernoulli random variable that comes from the following distribution:

$$\mu_e | (x_k, x_\ell) \sim \text{ Bern}\Big(q_e \mathbb{I}(x_k = x_\ell) \Big)$$

with $q_e = 1 - \exp(-\beta_{k\ell})$ for all $e \in E$. Therefore, it always holds that $\mu_e = 0$ and the edge is turned off when $x_k \neq x_\ell$. However, if $x_k = x_\ell$, then an edge between the vertices is "on" with probability q_e.

Using the information and notations from above, the Swendsen-Wang algorithm can be split into two main steps [2]:

(i) **The clustering step**

In a given current state, the edge $e = \langle k, \ell \rangle$ between vertices k and ℓ is always turned off if $x_k \neq x_\ell$. However, if their state variables are of the same value, the edge between vertices k and ℓ are turned on the with probability q_e. Having these series of edges being on and off will result in groups of vertices that are connected to one another by the "on" edges, but separate from vertices in other groups. These groups are known as connected components. Let V_i denote all the vertices that belong to component group i.

(ii) **The flipping step**

In this step, a connected component V_i will be selected randomly, and the corresponding state variable for all the vertices in V_i will be set to either 0 or 1 (determined with uniform

probability). The random label flipping can be repeated for one or all the connected components independently. Now that some state variables have changed, step (i) is repeated again.

To clear up some potentially confusing notations, it should be clarified that the edges in the Ising model are *not* the same as the ties connecting the nodes in the network. In the network, a tie can be between any two nodes and is represented in the adjacency matrix. Whereas in an Ising model, vertices are stuck in a lattice formation with their edges forming the lattice. Thus, an edge in the Ising model, denoted by $e = \langle k, \ell \rangle$, can only connect a vertex to its four nearest neighbors (above, below, left, and right) in the lattice and is only used to help allocate group membership in the clustering step to prepare for the flip. Rather, it is the state variable of each node in the $n \times n$ lattice that is used as elements of the adjacency matrix ($x_{k=i+j} = y_{ij}$).

A closer inspection of the Swendsen-Wang algorithm reveals many of its similarities to the Metropolis-Hastings algorithm.

Each time the clustering and flip steps are repeated, our Ising model ($\boldsymbol{Y}$) will switch from one state (denoted A), into another new state (denoted B). The acceptance probability for the Ising model to move from the current state A to the proposed state B is

$$p(A \to B) = \min\left(\frac{Q(B \to A)}{Q(A \to B)} \times \frac{\pi(A)}{\pi(B)}\right) \tag{4.2}$$

where the ratio of proposal probabilities is

$$\frac{Q(B \to A)}{Q(A \to B)} = \frac{(1-q_e)^{|C_{01}|}}{(1-q_e)^{|C_{02}|}} = (1-q_e)^{|C_{01}|-|C_{02}|} \tag{4.3}$$

where $|C_{ij}|$ is the number of edges that were "turned off" to separate connected component V_i from V_j when the $\boldsymbol{Y}$ (Ising model) moves from state A to stats B. Note that this proposal probability matches the term $\frac{q(\boldsymbol{y}_{\text{current}}, \boldsymbol{y}_{\text{proposed}})}{q(\boldsymbol{y}_{\text{proposed}}, \boldsymbol{y}_{\text{current}})}$ from the Metropolis-Hastings algorithm.

In addition,

$$\frac{\pi(A)}{\pi(B)} = \frac{e^{-\beta|C_{02}|}}{e^{-\beta|C_{01}|}} = e^{\beta(|C_{01}|-|C_{02}|)}. \tag{4.4}$$

Again, this is equivalent to $\frac{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y}=y_{\text{proposed}})}{\mathbb{P}_{\theta_0,\mathcal{Y}}(\boldsymbol{Y}=y_{\text{current}})}$ in the Metropolis-Hastings algorithm.

Finally, using the information from above, it can be seen that the probability in equation 4.2 can be rewritten as

$$p(A \to B) = \min\left(\frac{Q(B \to A)}{Q(A \to B)} \times \frac{\pi(A)}{\pi(B)}\right) = \left(\frac{e^{-\beta}}{1-q_e}\right)^{|C_{01}|-|C_{02}|} = 1. \tag{4.5}$$

Thus, we can see that this algorithm is essentially like a Metropolis-Hastings algorithm that always moves onto the proposed state.

4.1.1 Problems

While the Swendsen-Wang algorithm with Ising model is very similar to the Metropolis-Hastings algorithm and has faster mixing times, multiple simulations have revealed its difficulty in simulating undirected networks. This is likely due to the restrictive nature of the model's lattice formation. The edge restrictions (4 nearest neighbors) during the clustering step of the algorithm make it difficult to ensure that the state variables of all the vertices will have the appropriate values to form a symmetric adjacency matrix.

Additionally, similar to the Metropolis-Hastings algorithm, the Swendsen-Wang algorithm with Ising model also converges often to degenerate graphs. Once again, this problem seems to be mainly due to the lattice structure of the Ising model rather than the Swendsen-Wang algorithm itself.

4.2 Gibbs Sampler with Swendsen-Wang Method

In the previous section, most of the problems associated with using the Swendsen-Wang algorithm to simulate networks from ERGMs were due to the rigidity of the $n \times n$ lattice of the Ising model that only allowed for 4 nearest neighbor connections. A potential remedy for this could be to generalize the ERGM into an exponential-family random network model (ERNM) [4]. Along with the nodal ties of the network, ERNMs can also model nodal attributes (e.g., sex, age, group membership, etc.) as random variables and create a random model of the full network.

Let $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$ be the adjacency matrix as described in the ERGM. We introduce a new random variable $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ to be the matrix of nodal covariates where m is the number of nodal attributes in the network (e.g., sex, age, group, etc). This paper will only focus on one nodal attribute of the network at a time, and thus $\boldsymbol{X} \in \mathbb{R}^{n}$. $\boldsymbol{X}$ entries x_i indicate the nodal attribute category that the i-th node of the network belongs to (i.e., male or female for the attribute sex). Combining these two variables, let the network to be the random variable $(\boldsymbol{Y}, \boldsymbol{X})$. If we let the support of $(\boldsymbol{Y}, \boldsymbol{X})$ be $\mathcal{N}$, then the joint exponential family model becomes

$$\mathbb{P}_{\boldsymbol{\theta},\boldsymbol{\gamma},\mathcal{N}}(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{X} = \boldsymbol{x}) = \frac{\exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y}) + \boldsymbol{\gamma}^\top \boldsymbol{h}(\boldsymbol{x})\right)}{\kappa(\boldsymbol{\theta}, \gamma, \mathcal{N})}\,. \tag{4.6}$$

The ERNM model that we are interested in using in this section is similar to the Joint Ising/Potts model [4] but without the lattice and 4 nearest neighbor constraint. In this model, $\boldsymbol{g}(\boldsymbol{y})$ is the density of homophilous and non-homophilous ties in the network $\boldsymbol{y}$:

$$\boldsymbol{g}(\boldsymbol{y}) = \left[\frac{\#\text{ homopilous ties}}{\#\text{ homopilous dyads}}\,,\, \frac{\#\text{ non-homopilous ties}}{\#\text{ non-homopilous dyads}}\right]. \tag{4.7}$$

In turn, $\boldsymbol{h}(\boldsymbol{x})$ is the number of nodes in each nodal attribute category/group:

$$\boldsymbol{h}(\boldsymbol{x}) = [\#\text{ nodes in group 1 }, \#\text{ nodes in group 2}, ..., \#\text{ nodes in group } m\,]\,. \tag{4.8}$$

Simulations can be generated from this ERNM model by using the Gibbs sampler together with the Swendsen-Wang algorithm. This is done by seperately generating from the conditional distributions

$$\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x};\, \boldsymbol{\theta};\, \boldsymbol{\gamma}) \propto \exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})\right) \tag{4.9}$$

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{Y} = \boldsymbol{y};\, \boldsymbol{\theta};\, \boldsymbol{\gamma}) \propto \exp\left(\boldsymbol{\gamma}^\top \boldsymbol{h}(\boldsymbol{x})\right) \tag{4.10}$$

and using Gibbs sampling to iteratively use the results of one to generate the other.

For a given adjacency matrix $\boldsymbol{y}$, $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{Y} = \boldsymbol{y};\, \boldsymbol{\theta};\, \boldsymbol{\gamma})$ is generated using the Swendsen-Wang algorithm. In this model, however, the nodes are no longer restricted to forming ties with its four nearest neighbors. In addition x_i's are features of the nodes and entires of $\boldsymbol{X}$ rather than the values of the adjacency matrix like in Chapter 4. The clustering step takes the given $\boldsymbol{y}$ and clusters the nodes by grouping them into different connected components (recall the definition from section 2.3) based on the ties within this adjacency matrix. While connected components are usually reserved for undirected graphs, in this paper, nodes from directed graphs are considered connected if there is a path between them that consists of only reciprocated ties. The flipping step of the Swendsen-Wang algorithm takes each component and randomly assigns all the nodes in that component to a single nodal attribute category. This will generate an $\boldsymbol{x}$ consisting of the updated nodal attribute value for each node.

For a given nodal attribute matrix $\boldsymbol{x}$, $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x};\, \boldsymbol{\theta};\, \boldsymbol{\gamma})$ is a dyad independent model and thus each y_{ij} can be generated using the Bernoulli distribution

$$\text{Bern}\Big(\theta_1 \mathbb{I}(x_i = x_j)\mathbb{I}(i \neq j) + \theta_2 \mathbb{I}(x_i \neq x_j)\Big)$$

.

An additional benefit of utilizing this algorithm is that on top of simulating overall network ties, it can also simulate both homophilous and non-homophilous ties in the network now that nodal attributes of the networks are also introduced into the model.

4.2.1 Problems

While combining the Gibbs sampler with the Swendsen-Wang algorithm did solve several of the major issues mention in section 4.1.1, it also brought up a significant problem of its own. The majority of the time, after a few iterations of Gibbs sampling, the all nodes will start to be grouped into the same connected component and thus be assigned the same nodal attribute value. As a results, all the ties in the network will become homophilous ties.

This problem seems to be mainly due to the flipping step of the Swendsen-Wang algorithm, when the algorithm is trying to simulate an $\boldsymbol{X}$ given a $\boldsymbol{y}$. After clustering all the nodes into connected components, the flipping step assigns all the nodes in the same component with the same random attribute category. Since this is done randomly and without regard to the number of nodes within each connected component, the method could easily set a majority (and consequently in a few more iterations, all) of the nodes with the same attribute category. This would be even more likely to happen if the original network the algorithm is simulating from has a single large connected component to begin with. Once all the nodes are grouped into the same connected component, all the remaining iterations will most likely do the same, eventually causing the count of homophilous ties (and thus overall density of ties) to be much higher than that of the original network. As a result, this algorithm is unable to generate good simulations of a network.

CHAPTER 5

Iterative Sampling with Spectral Clustering Simulation

The only significant problem with the Gibbs sampler with Swendsen-Wang algorithm was in the step of sampling a new nodal covariate matrix $\boldsymbol{X}$ given an adjacency matrix $\boldsymbol{y}$ by using the Swendsen-Wang algorithm. In particular, its method of clustering the nodes and flipping its $\boldsymbol{x}$ values introduced a lot of problems. Thus this section will propose the iterative sampling with spectral clustering method instead, which keeps everything else the same from the method proposed in section 4.2, but switches out the Swendsen-Wang algorithm with an alternative way of updating $\boldsymbol{x}$ given $\boldsymbol{y}$.

The stochastic block model is used to generate networks consisting of groups and clusters such that nodes within each group are tied to one another with a particular density. As a result, nodes within the same cluster could be more likely to be tied to one another than with nodes outside of the cluster [1].

Spectral clustering can be applied to adjacency matrices by finding the eigenvalues of the matrix, performing dimensionality reduction, and clustering nodes into lower dimensions [1].

Therefore, using spectral clustering with stochastic block models on the adjacency matrix will group the nodes into different clusters (matching the number of categories in the nodal attribute) based on both the eigenvalues of the adjacency matrix and the pattern/densities of its ties.

Rather than using the Swendsen-Wang algorithm, the iterative sampling with spectral clustering algorithm will instead use spectral clustering in stochastic block models to cluster the nodes and update their nodal attribute values given $\boldsymbol{y}$.

Something important to note is that when using spectral clustering to generate $\boldsymbol{x}$ given $\boldsymbol{y}$, there is only one possible solution for the resulting $\boldsymbol{x}$. This eliminates the randomness in the generation of $\boldsymbol{x}$ given $\boldsymbol{y}$ (and the algorithm no longer involves Gibbs sampling). Thus, we switch from simulating from an ERNM model back into simulating from an ERGM model with specification

$$\mathbb{P}_{\boldsymbol{\theta},\mathcal{Y}}(\boldsymbol{Y} = \boldsymbol{y}) = \frac{\exp\left(\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{y})\right)}{\kappa(\boldsymbol{\theta}, \mathcal{Y})} \tag{5.1}$$

and where $\boldsymbol{g(y)}$ still follows the same homophily specifications from equation 4.3.

5.1 Algorithm

In Algorithm 1 below, we see the pesudocode for generating and updating $\boldsymbol{x}$ (non-random) and $\boldsymbol{y}$ (random). $\boldsymbol{x}$ given $\boldsymbol{y}$ is generated using spectral clustering in stochastic block models as described in the beginning of this chapter.

Recall from section 4.2 that because $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$ is a dyad independent model, $\boldsymbol{y}$ given $\boldsymbol{x}$ is generated using the distribution

$$\boldsymbol{y} \sim \text{Bern}\Big(\theta_1 \mathbb{I}(x_i = x_j)\mathbb{I}(i \neq j) + \theta_2 \mathbb{I}(x_i \neq x_j)\Big)$$

Algorithm 1 Iterative Sampling with Spectral Clustering: Updating X and Y

function UPDATEX($\boldsymbol{y}$)
    $clusters$ = SPHERICALSPECTRALCLUSTERING($\boldsymbol{y}$)
    $\boldsymbol{x}$ = MEMBERSHIP($clusters$) ▷ extract which cluster each node is in
    return $\boldsymbol{x}$
end function
function UPDATEY($\boldsymbol{x}$, directed, $\theta_{\text{non-hom}}$, θ_{hom})
    n= number of nodes
    for i = 1,...,n **do**
        for j=1,...,n **do**
            if $x_i = x_j$ **then** $y_{ij} = 0$
            else if $x_i \neq x_j$ **then** $y_{ij} \sim \text{Bern}(\theta_{\text{non-hom}})$
            else $y_{ij} \sim \text{Bern}(\theta_{\text{hom}})$
            end if
        end for
    end for
    if not directed **then** copy upper triangular matrix to bottom triangular matrix
    end if
    return $\boldsymbol{y}$
end function

Algorithm 2 below shows the code for the iterative sampling step which will call the `UpdateX` and `UpdateY` functions from Algorithm 1. We first generate an initial $\boldsymbol{x}_1$ (the only $\boldsymbol{x}$ that will be randomly generated) which assigns a nodal attribute category for each node in the network. While this first $\boldsymbol{x}_1$ is randomly generated, the distribution of the category densities within it should still be close to that of the originally observed network.

Once the $\boldsymbol{x}_1$ is generated, The function will iteratively use $\boldsymbol{x}_1$ in `UpdateY` to generate $\boldsymbol{y}_1$, which itself will then be used in `UpdateX` to generate $\boldsymbol{x}_2$, and so on.

Algorithm 2 Iterative Sampling with Spectral Clustering: Iterative Sampling

function ERGMSIM(net, $attr$, n, $\boldsymbol{\theta}$)
  $direct$ = ISDIRECTED(net)
  $catDens$ = DENSITY($attr$)
    ▷ densities of the categories of the nodal attribute $attr$ in net
  $\boldsymbol{x}$ = randomly generate a vector of categories of nodal attribute
    ▷ density of each category in $\boldsymbol{x}$ should match those of $catDens$
  for $i = 1, ..., 100 + n$ **do** ▷ generate n samples with 100 burn-ins
    $\boldsymbol{y}$ = UPDATEY($\boldsymbol{x}$, directed, θ_2, θ_1)
    $simNets[i]$ = MAKENETWORK($\boldsymbol{y}, \boldsymbol{x}$)
      ▷ combine nodes, attributes, and ties to make full network
    $\boldsymbol{x}$ = UPDATEX($\boldsymbol{y}$)
  end for
  $simNets = simNets[101 : (100 + n)]$ ▷ delete burn-ins
  return $simNets$
end function

A burn in of 100 iterations is used and all the $\boldsymbol{y}$'s generated after that will be simulations of the original network. Trying this iterative sampling with spectral clustering algorithm on existing networks have generated simulations with very accurate counts and densities of overall, homophilous, and non-homophilous ties. The details of this analysis will be explored further in the next chapter. However, this does mean that we can successfully sample networks from ERGMs using the iterative sampling with spectral clustering method.

CHAPTER 6

Analysis of Iterative Sampling with Spectral Clustering Simulations

In order to determine how well the iterative sampling with spectral clustering algorithm, which we can denote as ISSC for short, simulates networks from ERGMs, it will be used to simulate both a directed and undirected network. In order to determine the performance of the algorithm, we will analyze the density, homophilous ties, and non-hompilous ties of the simulated networks and compare them to that of the original observed networks.

6.1 Directed Networks

The directed network we will be looking at is the commonly studied Sampson network from the **ergm** package in R [6]. In the Sampson network, each of the 18 nodes represents a monk in a certain monastery. A directed tie from one node to another means that the first monk selected the second monk as one of three monks he liked the most. After analyzing the trends and patterns of the ties between nodes, Sampson [11] identified three main groups: Young Turks, Loyal Opposition, and Outcasts. The Loyal Opposition group consists of monks who joined the monastery first, the Young Turks arrived later when the monastery was undergoing a period of change, and the Outcasts are monks who were not accepted into either of the two groups. In the Sampson network, these endogenous group memberships are indicated for each node using the nodal attribute `group`. Examination of the networks' homophily (both the simulated and originally observed network) will be done using this

nodal attribute.

100 simulations were generated using the ISSC algorithm and the results are shown in the boxplot below with the red dots representing values observed in the original Sampson network. The parameters used in the function are the densities of homophilous and non-homophilous ties in the Sampson network, which are $\theta_1 = 0.1190476$ and $\theta_2 = 0.65625$ respectively.

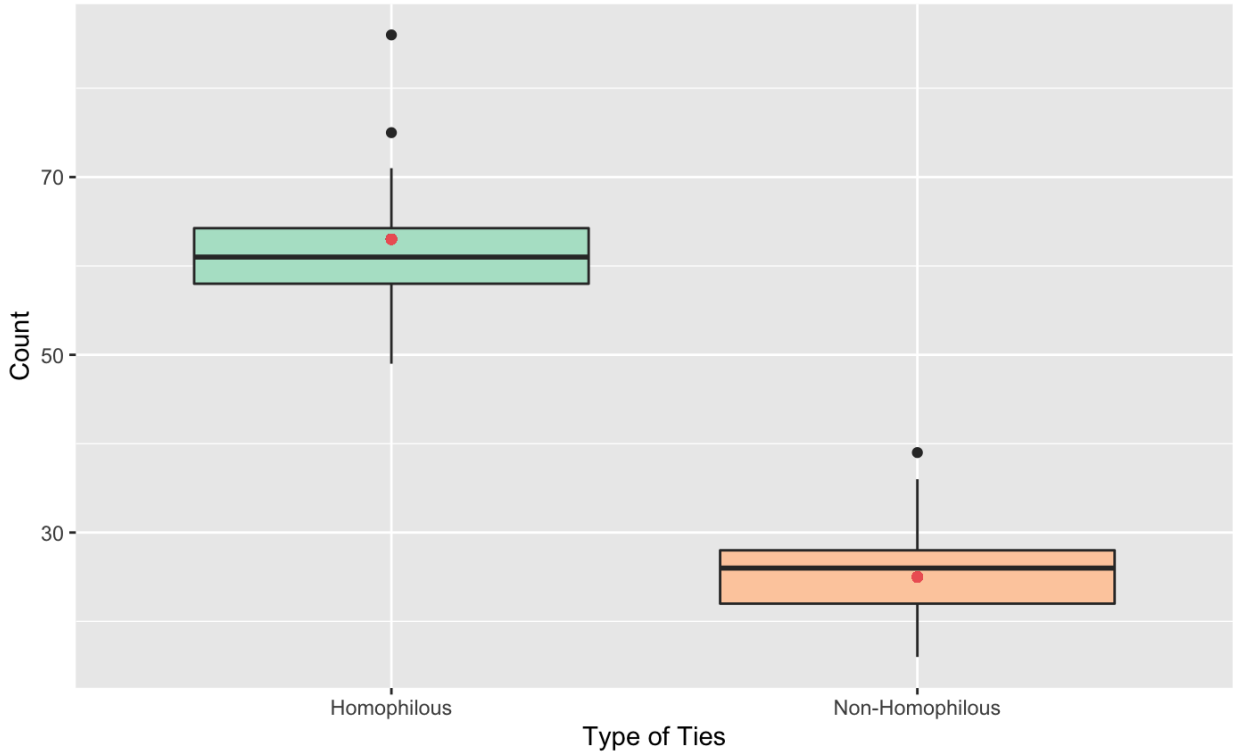

Figure 6.1: Boxplot showing counts of homophilous and non-homophilous ties of 100 networks simulated from the Sampson network using the ISSC algorithm.

We can see from Figure 6.1 that the observed count of homophilous ties from the Sampson network falls slightly below the 75th percentile of count of homophilous ties in the simulated networks. In addition, the observed count of non-homophilous ties is pretty close to the mean count of non-homophilous ties in the simulated network. Thus, it seems that the algorithm

did fairly well simulating homophily from the original network. Since overall number of ties in the network is simply a sum of the homophilous and non-homophilous ties, the algorithm also did fairly well simulating the overall density of ties of the Sampson network.

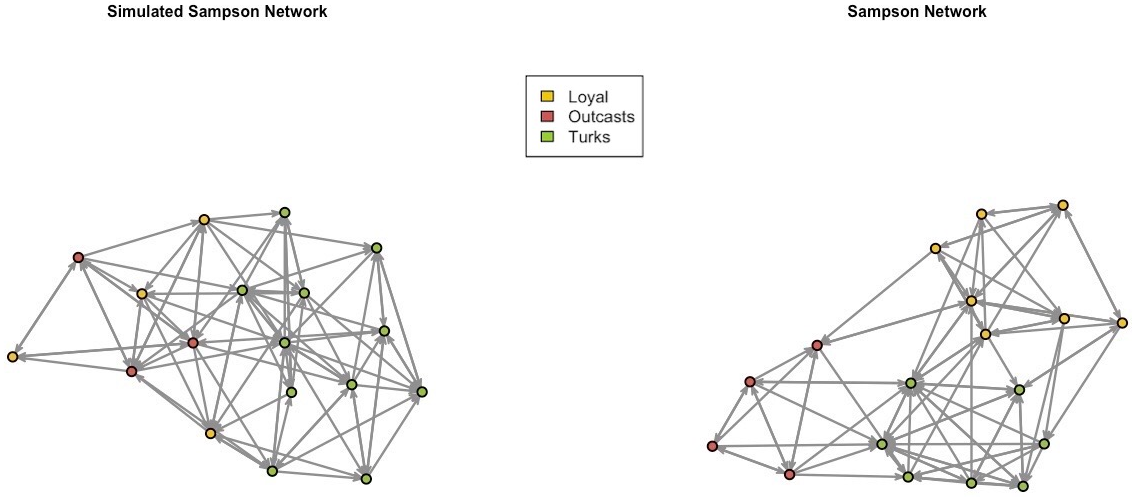

Figure 6.2: A simulated network (left) of the Sampson network (right) using the ISSC algorithm

One of the 100 simulated networks, as well as the Sampson network, are shown in Figure 6.2. The two networks are fairly similar in appearance. In addition, there are clustering of nodes into groups (Loyal, Turks, Outcasts) in the simulated network, just like in the original.

6.2 Undirected Networks

The undirected graph that we are simulating from is the Co-worker network from the Lazega law firm networks dataset [10]. Each of the 71 nodes in the network represents an attorney of a firm. An undirected tie between two nodes indicates that those two employees have worked together before at some capacity. While there are several nodal attributes to look at (gender, office, school, etc), Lazega [10] determined that `status` is a very important grouping

factor. The two categories for `status` are Partner and Associate. Thus, examination of the networks' homophily will be done using this nodal attribute.

100 simulations were generated using the algorithm and the results are shown in the boxplot below with the red dots representing values observed in the original Co-worker network. The parameters used in the algorithm are the densities of homophilous and non-homophilous ties in the Co-worker network, which are $\theta_1 = 0.1777778$ and $\theta_2 = 0.1257143$ respectively.

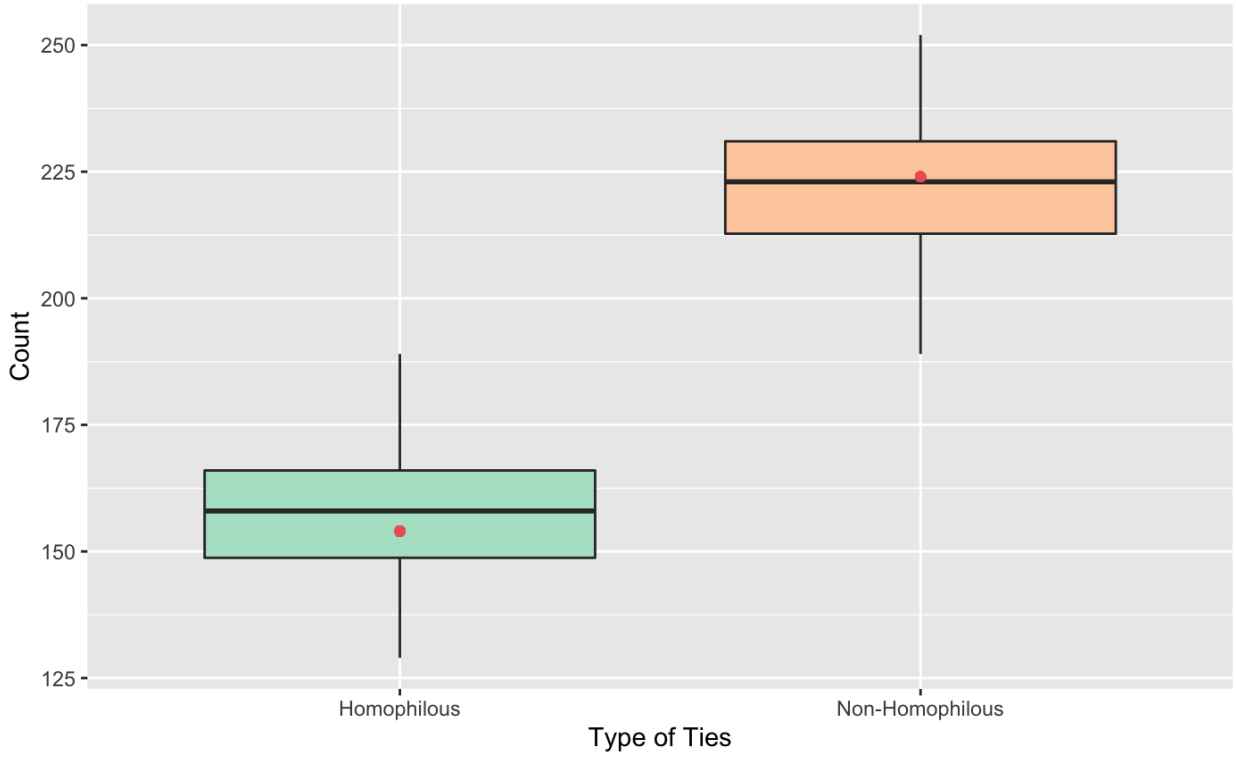

Figure 6.3: Boxplot showing counts of homophilous and non-homophilous ties of 100 networks simulated from the Co-worker network using the ISSC algorithm.

We can see from Figure 6.3 that the observed count of homophilous ties from the Co-worker network falls more than halfway between the 25th and 50th percentile of counts of homophilous ties in the 100 simulated networks. In addition, the observed count of

non-homophilous ties is extremely close to the mean count of non-homophilous ties in the simulated network. Thus, it seems that the algorithm did fairly well simulating homophily and overall tie density from the original undirected network as well.

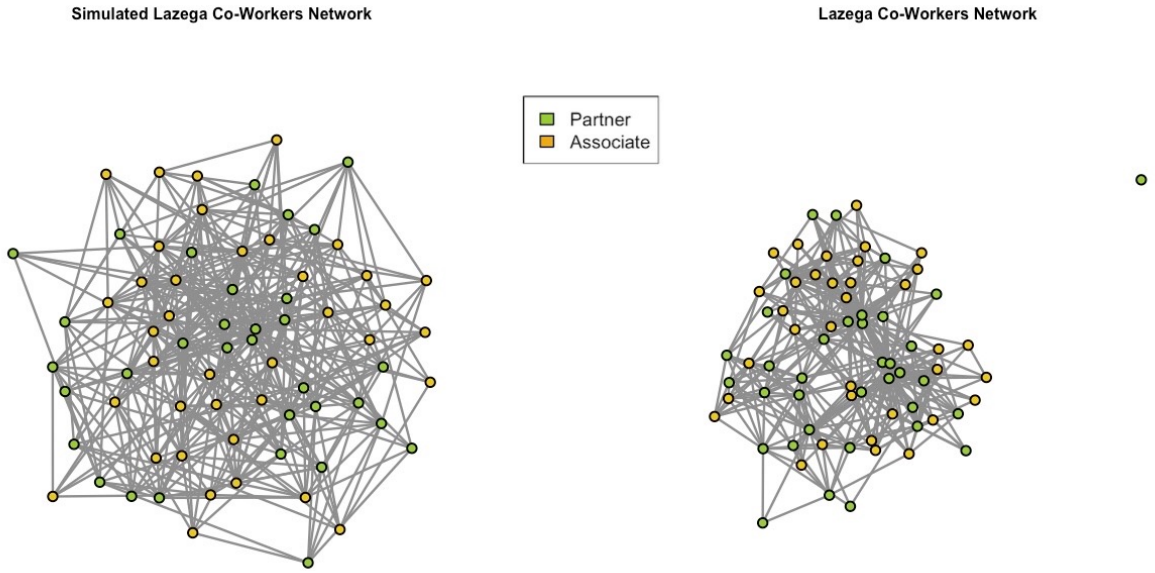

Figure 6.4: A simulated network (left) of the Co-worker network (right) using the ISSC algorithm

The Co-worker network and one of the simulated networks are shown in Figure 6.4. With exception the single unconnected node in the original network, the overall shape of the two networks are fairly similar in appearance. While the clustering of nodes into groups (Partner and Associate) is not as obvious as it was in the Sampson network, it is still present in both the simulated and Co-worker Network.

CHAPTER 7

Inference of Iterative Sampling with Spectral Clustering Simulations

Now that we have the ISSC algorithm to generate simulations from the ERGM, they can be used to look at inferences of the model. This chapter will focus specifically on using the ISSC method to look for the maximum-likelihood estimator (MLE) for the ERGM's $\boldsymbol{\theta}$ parameter. The simulations in Chapter 6 were generated using the density of homphilous and non-homphilous ties in the original observed network. The results of the boxplots in figure 6.1 and 6.3 show that the observed values of counts for homphilous and non-homphilous ties from the original networks are mostly close to the mean values of the simulated networks. Using the MLE of $\boldsymbol{\theta}$ should ideally bring the means of the simulated networks closer to the true observed value.

The algorithm to find the MLE of $\boldsymbol{\theta}$ is as follows:

Algorithm 3 Finding the MLE of $\boldsymbol{\theta}$

1: Generate 100 simulations of network $\boldsymbol{y}_n = \{\boldsymbol{y}\}_{n=1}^{100}$ using ISSC with $\boldsymbol{\theta'} = (\theta'_1, \theta'_2)$

2: Compute $\boldsymbol{g(y_n)}$ (recall equation 4.7)

3: Compute $\text{LS}(\boldsymbol{\theta'}) = \left(\overline{\boldsymbol{g_1(y_n)}} - \boldsymbol{g_1}(\boldsymbol{y}_{\text{obs}})\right)^2 + \left(\overline{\boldsymbol{g_2(y_n)}} - \boldsymbol{g_2}(\boldsymbol{y}_{\text{obs}})\right)^2$

▷ $\boldsymbol{y}_{\text{obs}}$ is $\boldsymbol{y}$ from the original network (Sampson, Co-work, etc.)

4: $\hat{\boldsymbol{\theta}}_{MLE} = \min_{\boldsymbol{\theta'}} \text{LS}(\boldsymbol{\theta'})$

7.1 Directed Networks

The Sampson network will once again be used as the directed network example. Using Algorithm 3, the MLE of $\boldsymbol{\theta}$ in ERGM used to simulate the Sampson network is $\hat{\boldsymbol{\theta}}_{MLE} = (0.1152947, 0.6590157)$.

Compared to the $\boldsymbol{\theta}' = (0.1190476, 0.65625)$ used for simulating the Sampson network from Chapter 6 (which are simply the densities of homophilous and non-homophilous ties in Sampson), it is evident that the two are almost the same. This is to be expected as the simulations generated from $\boldsymbol{\theta}'$ were already fairly accurate.

An updated boxplot of 100 simulations generated from using $\hat{\boldsymbol{\theta}}_{MLE}$ in the ISSC algorithm is shown in figure 7.1.

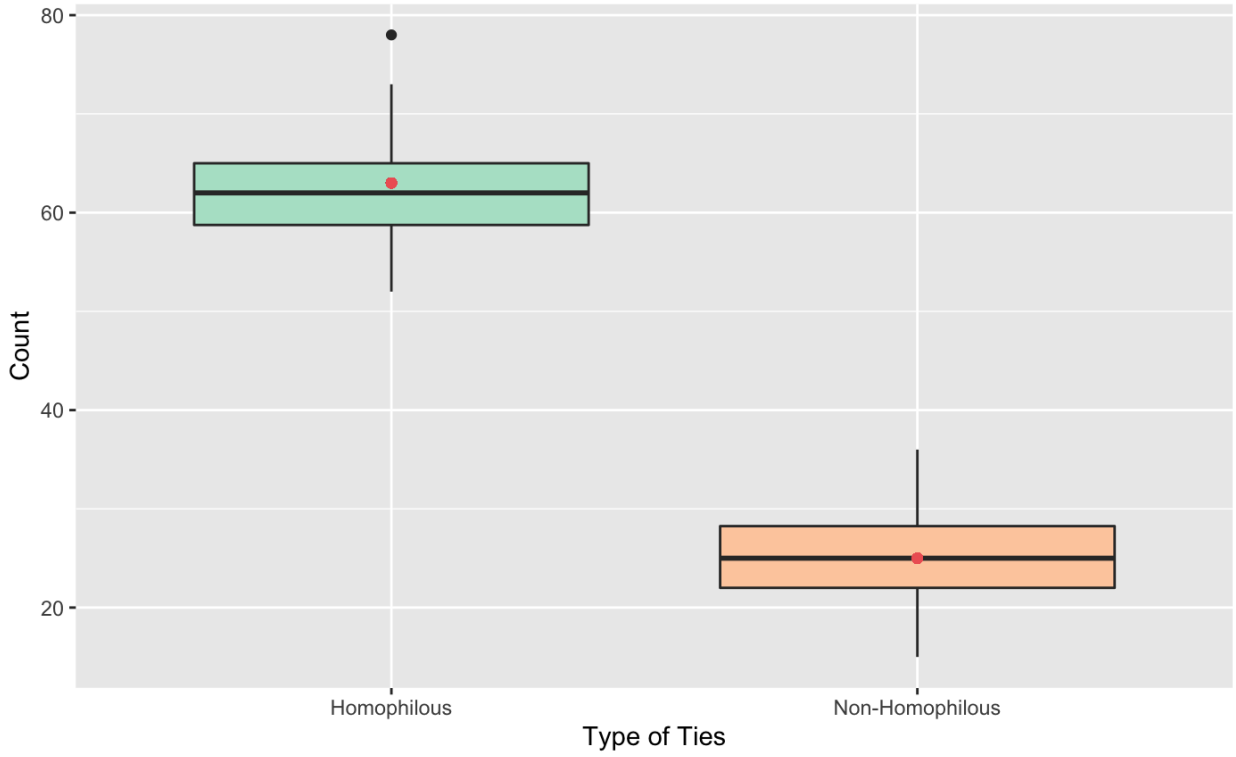

Figure 7.1: Boxplot showing counts of homophilous and non-homophilous ties of 100 networks simulated from the Sampson network using $\hat{\boldsymbol{\theta}}_{MLE}$.

Figue 7.1 shows that by using $\hat{\boldsymbol{\theta}}_{MLE}$, the observed count of homophilous and non-homophilous ties matches almost exactly to the mean count of homophilous and non-homophilous ties of the 100 simulated networks. Thus between $\hat{\boldsymbol{\theta}}_{MLE}$ and $\boldsymbol{\theta}'$, the MLE is the better parameter to use in the simluation of Sampson using the ISSC algorithm.

7.2 Undirected Networks

For the Co-worker network, the MLE of $\boldsymbol{\theta}$ is $\hat{\boldsymbol{\theta}}_{MLE} = (0.1776876, 0.1256337)$. Comparing it to the $\boldsymbol{\theta}' = (0.1777778, 0.1257143)$ used for simulating the Co-worker network from Chapter 6 (which, again, are the densities of homophilous and non-homophilous ties), it's evident that they are extremely similar.

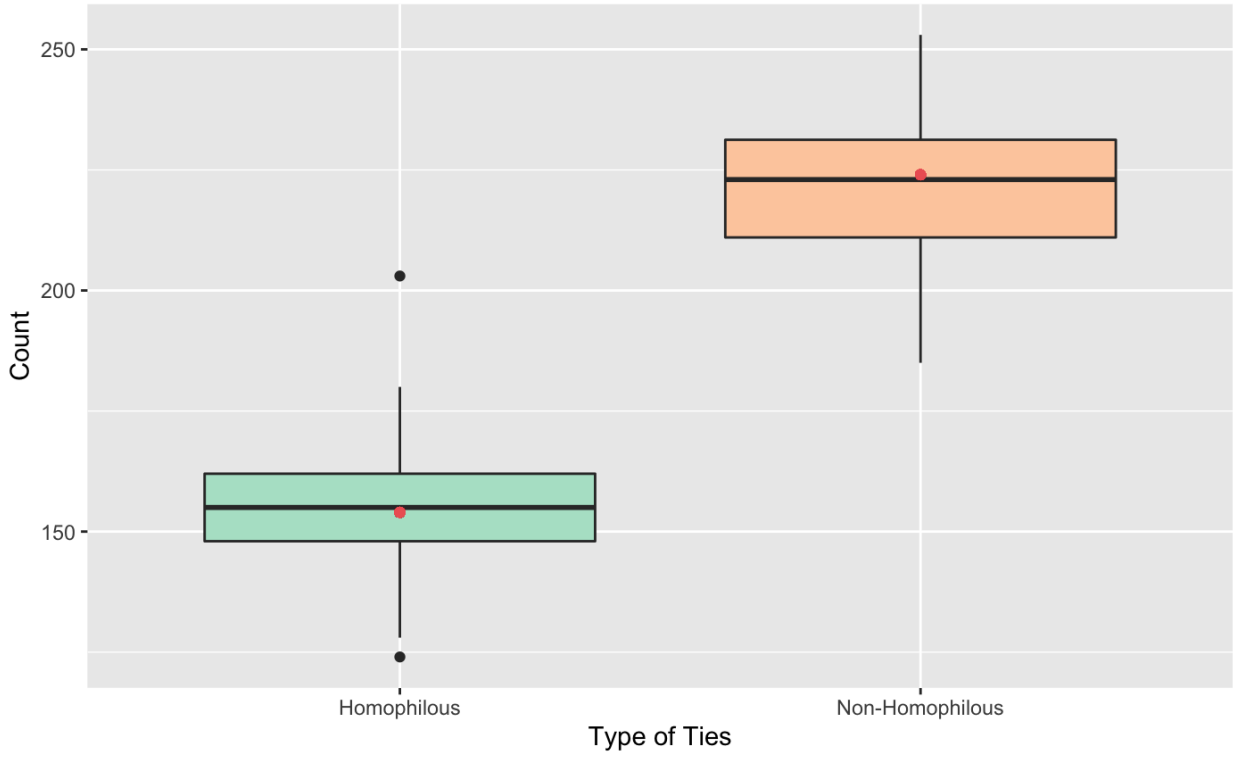

Figure 7.2: Boxplot showing counts of homophilous and non-homophilous ties of 100 networks simulated from the Co-worker network using $\hat{\boldsymbol{\theta}}_{MLE}$.

Figue 7.2 shows that by using $\hat{\boldsymbol{\theta}}_{MLE}$, the observed count of homophilous and non-homophilous ties matches almost exactly to the mean count of homophilous and non-homophilous ties of the 100 simulated networks. Thus between $\hat{\boldsymbol{\theta}}_{MLE}$ and $\boldsymbol{\theta}'$, the MLE is the better parameter to use in the simluation of Co-worker using the ISSC algorithm.

CHAPTER 8

Conclusion

After trying to simulate networks from ERGM/ERNMs using the Swendsen-Wang algorithm, the Gibbs sampler with Swendsen-Wang algorithm, and the iterative sampling with spectral clustering algorithm, it is evident that the ISSC method is the only viable method of out of the three.

The Swendsen-Wang algorithm with Ising model is very similar to the commonly used Metropolis-Hastings algorithm but without the potential issues of slow mixing time and degeneracy. However, due to the rigidity of lattice formation in the Ising model, it failed to accurately simulate from ERGMs.

The Gibbs sampler with the Swendsen-Wang algorithm solved many of the major problems in the previous model and introduced the possibility of simulating homophily on top of overall network density. However, due to issues with the flipping step of its Swendsen-Wang portion of the algorithm, it also failed to provide accurate simulations from ERMNs.

The ISSC algorithm replaces the problematic Swendsen-Wang step from the previous method with spectral clustering in stochastic block models. After testing the method on both a directed and undirected network, it was able to simulate homophily and overall tie densities of a network very accurately. Thus, we could conclude that the ISSC could be an alternative to the Metropolis Hastings algorithm for simulating networks from ERGMs. In addition, ISSC was able to perform inference on the ERGM by finding the MLE of its parameter $\boldsymbol{\theta}$. Further studies on this algorithm can look for ways expand its abilities to simulate other aspects of a network, such as the n-stars or triad census as well.

REFERENCES

[1] Abbe E. Community Detection and Stochastic Block Models: Recent Developments. *Journal of Machine Learning Research* (18): 1-86, 2017.

[2] Barbu A. and Zhu S. *Monte Carlo Methods*. Springer, Draft 2019.

[3] Cranmer S.J. and Desmarais B.A. Inferential network analysis with exponential random graph models. *Political Analysis*, 19(1):6686, 2011.

[4] Fellows, I. and Handcock M. Exponential-family Random Network Models. *arXiv: Methodology*, 2012.

[5] Frank, O. and Strauss D. Markov Graphs. *Journal of the American Statistical Association*, 81(395):832-842, 1986.

[6] Goodreau S.M., Handcock MS, Hunter DR, Butts CT, Morris M. A statnet Tutorial. *J Stat Softw*, 24(9):1-27, 2008.

[7] Goodreau S.M., Kitts J.A. Morris M. Birds of a feather, or friend of a friend? Using exponential random graph models to investigate adolescent social networks. *Demography*, 46(1):103-125, 2009.

[8] Handcock M. Assessing degeneracy in statistical models of social networks. *Journal of the American Statistical Association*, 76:33-50, 2003.

[9] Hunter D.R. and Handcock M.S. Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, 15(3):565-583, 2006.

[10] Lazega E. *The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership*. Oxford University Press, 2001.

[11] Sampson F.S *A Novitiate in a period of change: An experimental and case study of social relationships*. PhD thesis, Cornell University, 1968

[12] Saul Z.M. and Filkov V. Exploring biological network structure using exponential random graph models. *Bioinformatics*, 23(19):2604-2611, 2007.

[13] Swendsen R. and Wang J. Nonuniversal Critical Dynamics in Monte Carlo Simulation. *Physical Review Letters*, 58(2):86-88, 1987.